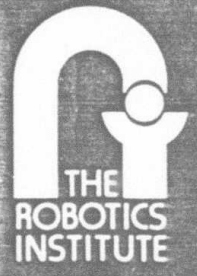
AD-A184 328

(12)

LISP as a Rapid Prototyping Environment: The Chinese Tutor

Dario Giuse

CMU-RI-TR-87-14



Carnegie Mellon University

The Robotics Institute

Technical Report

LISP as a Rapid Prototyping Environment: The Chinese Tutor

Dario Giuse

CMU-RI-TR-87-14

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

June 1987

This research was sponsored by the Defense Advanced Research Projects Agency (DoD), ARPA Order No. 4976, monitored by the Air Force Avionics Laboratory under contract F33615-84-K-1520. The views and conclusions are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD-A184 32

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CMU-RI-TR-87-14	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) LISP as a Rapid Prototyping Environment: The Chinese Tutor		5. TYPE OF REPORT & PERIOD COVERED Interim 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Dario Giuse		8. CONTRACT OR GRANT NUMBER(s) DoD ARPA Order No. 4976 AFL F33615-84-K-1520
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie Mellon University The Robotics Institute Pittsburgh, PA 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE June 1987 13. NUMBER OF PAGES 14
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Air Force Avionics Laboratory		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Approved for public release; distribution unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Conventional wisdom has maintained that while LISP is suitable for the development of experimental software, it does not support production-quality systems well. This report describes a building-block approach where a complex system (an intelligent Chinese language tutor) was built in a LISP environment out of a series of powerful tools. The resulting system does not in any way sacrifice performance for power and flexibility.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Table of Contents

List of Figures

Abstract

Conventional wisdom has maintained that while LISP is suitable for the development of experimental software, it does not support production-quality systems well. This report describes a building-block approach where a complex system (an intelligent Chinese language tutor) was built in a LISP environment out of a series of powerful tools. The resulting system does not in any way sacrifice performance for power and flexibility.

1. Introduction

LISP is one of the oldest programming languages [McCarthy 60], but unlike most languages of comparable age it has experienced a tremendous growth in the last few years. While older LISP systems were confined to the academic or research community, the introduction of Common Lisp [Steele 84] has brought the language to a much wider audience.

At the same time, recent hardware developments have made it possible to execute LISP efficiently on a number of machines. It is now possible to run very large LISP systems on workstations priced under $10,000. The acceptance of Common Lisp as a *de facto* industry standard and its availability on a wide range of machines have generated tremendous interest in the rapid development of LISP systems.

In this report I discuss the issue of rapid development of LISP software and I present as a concrete example a system written in Common Lisp. This system uses a layered approach to maximize flexibility while minimizing development time. My main goal was to create a useful tool in the shortest possible time. The major constraint was that the final product had to be sufficiently powerful and efficient to be used on a day-by-day basis.

The system, called the *Chinese Tutor*, is a language tutor for teaching beginner-level Chinese. It is highly interactive and supports a large database of Chinese characters along with the knowledge necessary to provide intelligent tutoring. In order to facilitate effective learning of Chinese, it contains an adaptive component that monitors the student's performance and concentrates tutoring on the areas where the student is weakest. A separate feature allows the student to use the system as a comprehensive online dictionary. Finally, the system contains facilities for maintaining its own database and customizing its user interface.

The first section of this report describes the Chinese Tutor (which I will refer to here as *Tutor*) from the user's point of view. Its main purpose is to provide some insight into the system's functionality, complexity, and performance.

The second section discusses the system's implementation and internal structure. It briefly describes the tools that were combined to build the system, with special emphasis on how the LISP environment facilitated their integration. It also contains an evaluation of how the different tools are used and how much they contribute to performance and memory usage.

The third section presents my overall experience in building the system and what lessons I have learned from the exercise. This section discusses advantages and disadvantages of the building-block approach and how it influences ease of development and maintenance. Finally, the last section briefly discusses future directions.

2. The Chinese Tutor

I will begin by describing the goals of the Chinese Tutor, since they have had a significant influence on the design and evolution of the system.

2.1 Goals

The main goals of the Tutor were:

- to provide immediate, effective support for a beginning student of Chinese (specifically, the author),
- to investigate the feasibility and performance of a building-block approach for a tutoring system, and
- to build a "real" system with a highly responsive interactive interface.

The first goal was instrumental in determining the choice of environment and programming language. Long development times were out of the question: I wanted to have a system running within three weeks, rather than three months. Common Lisp was the natural choice, being the only system that provides both complete interactive facilities and good performance on a personal workstation.

The building-block approach also stemmed quite naturally from the nature of the problem, which requires an *interactive component*, a *graphical component*, flexible *knowledge representation*, and *adaptability*. These considerations, together with my previous interest in the building-block approach (see, for instance, [Giuse *et al.* 83]), lead me to build the Chinese Tutor out of pre-existing tools unified by Common Lisp.

In order to cover the rapidly evolving needs of a beginning student, I designed the system to be very flexible. Rather than hard-wiring decisions into code, I used the knowledge representation system to store those decisions as procedural knowledge. Changing the behavior of the program could then be achieved simply by changing the knowledge base.

A secondary goal in the development of the Tutor was to validate and improve some of the tools developed as part of the Dante project. The Dante project [Giuse 86], currently underway at Carnegie Mellon University, investigates user interface systems issues. The main emphasis of the project is on the creation of what we call Uniform Workstation Interface, which will provide an integrated interface system for a highly heterogeneous, distributed computing environment.

Many tools used in constructing the Chinese Tutor were originally developed as part of the Dante project; their availability is one of the reasons for the Tutor's short development time. The Tutor has in turn provided a useful testbed for evaluating those tools and has thus given the Dante project significant feedback.

2.2 Functionality

The Chinese Tutor provides the following functionality:

- an interactive Chinese dictionary,
- documentation about the major elements of the Chinese language (characters, radicals, pronunciation, etc.), and
- an interactive Chinese tutor.

2.2.1 Interactive Dictionary

The dictionary allows a student to examine various aspects of Chinese characters or combinations of characters. The system is quite flexible and provides interactive equivalents of all the functions provided by paper dictionaries.[1] It is possible, for instance, to look up a character in several ways: by pronunciation, complete with tone marks; by approximate pronunciation, if the tone mark is not known; by radical; by number of strokes; and by meaning.

词 cí: word, speech. Radical is 言 讠 : speech, word, talk, character. [词典]: dictionary.

话 huà: word, talk, talk about. Radical is 言 讠 : speech, word, talk, character. [中国话]: spoken Chinese, Chinese. [说话]: say, chat, gossip. [白话]: vernacular language.

言 yán: speech, word, talk, character. Radical is 言 讠 : speech, word, talk, character. [语言]: language. [文言]: literary style, classical style.

Figure 2-1: A few entries in the dictionary

Figure 2-1 shows part of the output when the student requests to see all the entries for the English word "word." The portion of output illustrated in the figure shows three sections, each corresponding to a different Chinese character. Entries are shown according to the format commonly used in Chinese-English dictionaries: the Chinese character is shown first, followed by its pronunciation and tone mark, followed by the meanings. The radical for the character is indicated next, followed by the most important multi-character combinations with their translation. The entry for the character *ci*, for instance, shows the radical (which has two possible forms) and one important combination, namely, *ci-dian*, which means "dictionary."

Besides information about single characters, the system contains a substantial number of multiple-character entries.[2] Multiple-character entries can also be looked up by meaning, individual components, radicals, and pronunciation, just like single-character entries.

2.2.2 Online Documentation

The online documentation module offers a simple tutorial covering the basic Chinese language concepts. This tutorial is implemented in the VIEWERS system, which is described in the next section of this report. Entries in the tutorial explain fundamental concepts such as glyph, radical, and pronunciation. Examples of each concept are available. I have also planned, but not yet implemented, a simple description of the Chinese grammar.

Figure 2-2 shows the top level portion of the tutorial that teaches how to type a Chinese pronunciation. This is a typical VIEWERS system frame: it contains a title, some information, and some options. The reverse-video line at the bottom represents the "global pad," that is, options that are available from anywhere in the network.

A separate portion of the online documentation describes the Tutor itself. Using this portion the student

---

[1]The system, in fact, provides more extended functionality than traditional dictionaries, such as looking up a word by any portion of its translation or pronunciation.

[2]Even though Chinese characters are words by themselves, many words are formed by combining two or more characters to modify their meaning somewhat.

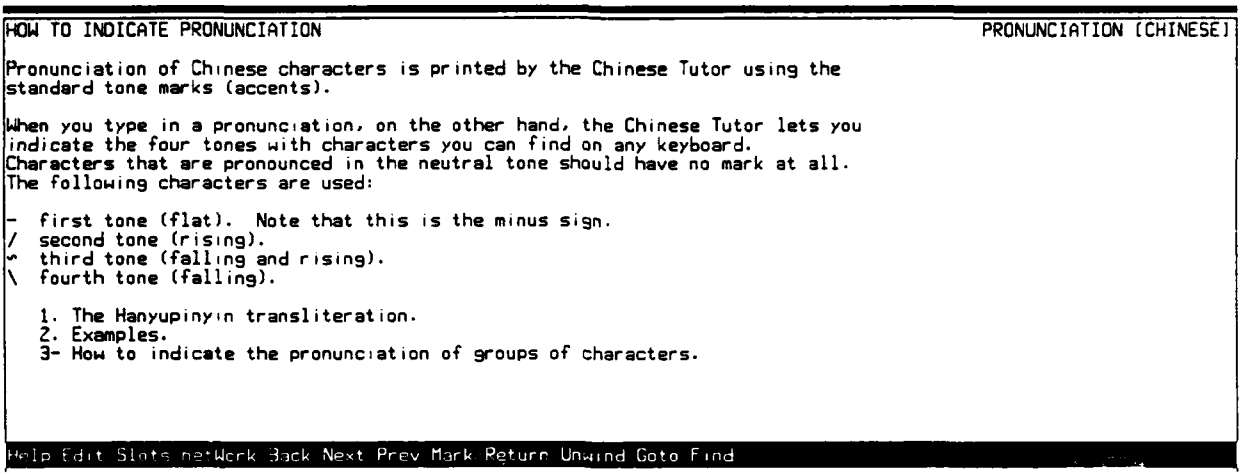


Figure 2-2: The frame that describes how to indicate pronunciation

can learn about the system and interactively try some of its basic options. Documentation about the user interface mechanism is also available.

2.2.3 Interactive Tutor

The interactive tutor is probably the most interesting portion of the Chinese Tutor. It not only tests the student's knowledge of Chinese, but in doing so it learns about the student's strengths and weaknesses and assists in those areas where the student is weakest.

The main idea is to ascertain how familiar the student is with the basic language components. To test familiarity, the Tutor may for instance:

- show the user a Chinese character and ask for its pronunciation, including tone,
- show the user an English word (i.e., a meaning) and ask for the pronunciation of the corresponding Chinese character or multi-character group, and
- show the user a Chinese character and ask what the corresponding radical is, or how many strokes are in the character.

Each test is initially random, that is, the Tutor will select among all questions with equal probability. As testing continues, the Tutor remembers how well the student did on each question and modifies the probabilities for each character. After a while, characters that are "well known" will be tested infrequently, whereas characters with a high error score will be asked much more often. Characters introduced only recently have a higher initial probability of being tested than characters that have been in use for longer periods of time.

In order to avoid long-term bias, older scores become less and less important. Once the student has learned a character "well," the character will be asked seldom, even if it was troublesome in the beginning. Even well-known characters, however, are tested every now and then as a way of reinforcing the student's recall.

Student performance scores are permanently saved with the database at the end of each session, so that the system effectively "remembers" the student's performance between sessions and adjusts its actions accordingly. An interactive command allows the student to correct simple mistakes (such as typos) without affecting the error scores.

The error scoring mechanism is not very sophisticated. In particular, it applies directly only to the more mechanical portions of learning Chinese. The Chinese Tutor does not currently incorporate some of the more typical aspects of intelligent tutoring systems as described, for instance, in [Carbonell 70]. While questions for the student are indeed generated from the internal knowledge base, rather than being predefined once and for all, the Tutor currently does not have any notion of curriculum and the student is responsible for selecting what topic to work on next.

Given the very memory-intensive nature of the task, however,[3] the current mechanism is quite effective in practice and covers a significant portion of the initial learning period. I am planning future extensions to the system that will improve its effectiveness by incorporating more structured notions of tutoring goals and by explicitly representing what *types* of errors the student made, so that remedial actions can be taken to correct basic student misconceptions.

2.3 Notation

The notation used in the Chinese Tutor corresponds to that found in modern textbooks and dictionaries. Chinese characters are displayed either in their complex form (corresponding to classical Chinese and currently used in Taiwan) or in their simplified form (introduced by the People's Republic of China in 1956). The student can choose at any time which form he or she wishes to see.

The pronunciation of Chinese characters is consistently indicated in *Hanyu Pinyin*, the modern system of transliteration which has essentially replaced the older Wades-Giles romanization. Tone marks[4] are displayed to the student as different types of accents, following standard usage, and are input on the keyboard as simple one-key characters.

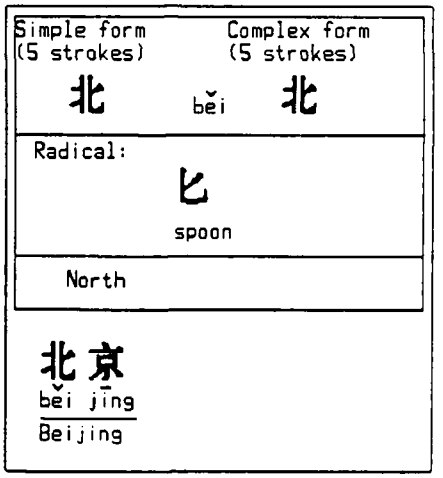


Figure 2-3: A sample character description

Figure 2-3 shows the entry for the character *bei* (North). The character is shown at the top in its simple and complex forms, which in this case are identical. Above each form is the number of strokes, which is

[3]In order to read Chinese fluently, the student should know about 4,000 Chinese characters. At least 800 characters are required for beginner-level understanding of written Chinese.

[4]Unlike Indo-European languages, Chinese is a tonal language and uses a specific voice tone for each character. The tone is an integral part of the spoken sound and is essential in determining the meaning of a character. A character can be pronounced in one of four tones, or short and without intonation (neutral tone).

often important when consulting paper dictionaries. Between the two forms of the character is the *Hanyu Pinyin* transliteration of the pronunciation. Figure 2-3, for instance, indicates that the character *bei* is pronounced in the third tone, as shown by the tone accent above the letter *e*.

The middle portion of the figure shows the radical for the character. Each written character has a radical, that is, a simpler character that appears in the written form and typically determines the meaning or the sound of the character. In our example, the radical for *bei* is the simpler character *bi*, which by itself means "spoon" or "ladle."

Under the radical comes the list of English translations for the character, which in figure 2-3 simply contains "North." Finally, the bottom portion of the figure shows the main compounds that contain the character. A prominent compound for *bei* is *bei jing*, literally "Northern capital," which is the Chinese name for Peking.

3. Implementation

The Chinese Tutor is currently implemented on the IBM RT/PC workstation. It runs under Mach [Tevanian 86] [Accetta *et al.* 86], an operating system developed at Carnegie Mellon University and based on Berkeley UNIX 4.2/4.3. The program is written in CMU Common Lisp, a version of Common Lisp running under Mach.

Following the building-block approach, I used existing tools whenever possible, rather than re-implementing their functionality. All of the tools I used are also implemented in Common Lisp. I will first briefly introduce the tools and then describe the structure of the Chinese Tutor itself.

3.1 The Tools

The Chinese Tutor uses the following tools: *Hemlock*, a text editor; *KR*, a knowledge-representation system; *Vector*, a vector-font display package; and *Viewers*, a frame-based browser and user interface system.

Hemlock [MacLachlan 86] is a Common Lisp version of the popular Emacs screen editor. Originally inspired by the Zmacs editor [Weinreb and Moon 81], it allows a user to customize it by writing LISP programs [MacLachlan 84]. Hemlock is not used directly by the Chinese Tutor, but it provides the basic support for the VIEWERS system which constitutes the main user-interface portion of the Tutor.

KR [Giuse 87] is a knowledge-representation system developed as part of the Dante project. KR represents knowledge as a semantic network. Nodes in the network are data structures called *schemata* and contain named attribute-value pairs. KR was influenced by SRL [Wright and Fox 83] [Fox *et al.* 84] and by the CRL language [Carnegie Group 86]; unlike those systems, however, it only provides the simplest layer of knowledge representation. The omission of the more complicated features for manipulating knowledge is compensated by very good performance, which makes KR suitable for user-interface environments requiring fast response time.

The **Vector** package provides a simple capability for displaying vector fonts. It represents fonts as sequences of short vectors, as opposed to the more conventional bitmaps. The vector format makes it possible to scale and rotate individual characters. The Vector package is implemented in Common Lisp and interfaces to the host machine's graphics system. The Tutor uses the Vector package in conjunction with the Hershey Oriental character set [Hershey 76], which contains several hundred Chinese characters.

Viewers is a user-interface tool and database manager. The interface is organized around a network of displays, called frames, that the user can freely traverse. Each frame may contain data, links to other frames, and possibly actions to be executed; see figure 2-2 for an example of a frame. VIEWERS was inspired by ZOG [Robertson, McCracken, and Newell 80], a text-based system developed at Carnegie Mellon University as a vehicle for experimentation in user interface issues. It was also partially inspired by NoteCards [XSIS 85], a system developed by Tom Moran at Xerox which allows each node in the network to contain both text and graphics.

3.2 Knowledge Representation

Knowledge in the Chinese Tutor is represented in the KR language. Knowledge can be divided into two parts: a fixed portion and a variable portion.

The *fixed portion* describes the basic elements of the Chinese language that are used by the Tutor and their inter-relationships. This portion contains, for instance, generic templates for a Chinese character, a radical, a glyph, etc. Facts like the existence of simple and complex forms for many Chinese characters,

or the conceptual relationship between Chinese radicals and their derived characters, are also represented in the fixed portion as schemata.

The fixed portion of knowledge is built into the Tutor and is immediately available at start-up time. It cannot be directly manipulated by the user, and therefore the Tutor does not contain a user interface to it.

The *variable portion* implements the Chinese characters database. It uses the fixed portion to describe relationships among actual characters and their sound, meanings, and printed representation. Figure 3-1 shows the schema representation for the radical *yi* and for the glyph used to display it.

```
{{T403
  is-a: :CHINESE-RADICAL
  radical-for: T31 T102 T348 T101 T326 T338 T344 T315
  meaning: "one"
  pronunciation: :YI-
  both-forms: T315
  }}

{{T315
  is-a: :GLYPH
  radical: T403
  glyph-for: T642 T403
  strokes: 1
  number: 1
  file: :CHI24
  type: :SIMPLE-AND-COMPLEX
  }}
```

Figure 3-1: Two schemata from the variable portion of the database

The schema T403[5] describes the radical *yi*, which means "one" and occurs in several characters. Its *pronunciation* slot is really a reference to another schema, YI-, which contains all the information about that particular sound.

Schema T403 is linked to the CHINESE-RADICAL schema through the *is-a* relation. It is also linked to a glyph schema, T315, through the *both-forms* relation. This relation, defined in the fixed portion of the database, indicates that the simple form and the complex form of a glyph are identical. The glyph schema, T315, points back to the radical and to another character schema (T642). It also indicates that the character is displayed by character number 1 in font file CHI24.

The variable portion of the database is the one that actually determines what characters are known to the system. It is stored in a simple database file, and the Chinese Tutor contains an extensive interface that lets the user inspect and modify the database at will. The variable portion of the knowledge base is also used to store information about the scoring mechanism. The latter is updated automatically as the student uses the Tutor and is saved to the external database at the end of each session.

3.3 The Program

The program that implements the Chinese Tutor and ties together the different components of the tool-kit is written in Common Lisp. It contains several sections:

- *Schema definitions* define the prototypical schemata in the system (such as a glyph in the simple form, a group of Chinese characters, etc.). They also define the KR relations used by

[5]Most of the schema names in the system are machine-generated.

the Chinese Tutor, such as the *both-forms* relation mentioned above.

- *Low-level functions* include schema manipulation, interface to the Vector package, low-level font access, network bookkeeping, and debugging aids.
- *Domain-specific functions* implement operations on Chinese objects and contain some knowledge about those objects. These are fairly generic operations, such as glyph manipulation and handling of the pronunciation of Chinese characters.
- *Scoring functions* implement the scoring mechanism and its interface to the database manager. These functions are completely domain-independent.
- *High-level functions* constitute most of the Tutor's user interface. They maintain and display the relationships among Chinese characters, radicals, groups, dictionary entries, and so on.
- *Database manager*: a simple database manager for saving and retrieving the current status of the Chinese knowledge base. It uses a machine-generated Common Lisp program as the external representation of the database.
- *Query functions* prompt the user for a given glyph, character, radical, group, sound, or meaning. These functions use the VIEWERS system and the Hemlock text editor.
- *Testing functions* interact with the VIEWERS system to evaluate the student's command of Chinese.

3.4 Statistics about the Chinese Tutor

I will now describe the Chinese Tutor in terms of source code size, object code size, and time to develop. The numbers consistently indicate that the building-block approach can indeed produce systems that are both relatively small and efficient. Even more important, the building-block approach combined with the flexibility of the LISP environment results in extremely short development time.

The program consists of 2,100 lines of LISP source code, organized into 150 functions. This is all the code that was hand-written to implement the Chinese Tutor. The combined size of the tools used by the Tutor, by comparison, is about 30,000 lines of source code.

The network that implements the user interface of the system consists of an additional 900 lines of LISP code, machine-generated by the VIEWERS system. The network contains 30 display frames and a total of 79 actions. The network is a graph closely resembling a shallow tree, with a maximum depth of 4 frames. The size of the network is independent of the number of Chinese characters actually defined by the database.

The external database is also machine-generated, and its size obviously depends on how many Chinese characters are defined. The current version contains about 600 lines of code. This is still a fairly small database, since it defines about 220 characters out of at least 4,000. It does, however, define all of the Chinese radicals and about 100 combinations (i.e., multiple-character entries). Once read into the program, the database corresponds to about 1,000 KR schemata.

Finally, some figures about the run-time size of the system. The VIEWERS system, including the system network, accounts for 204 Kbytes. The network for the Chinese Tutor adds another 86 Kbytes. The program and the database together account for 567 Kbytes. The whole system, therefore, requires 857 Kbytes at run-time on an IBM RT/PC. On a machine with greater code density this should correspond to about 500 Kbytes, which seems like a fairly reasonable size for a program of this complexity.

3.5 Development Time

The total time to develop the Chinese Tutor was two and a half man-weeks, distributed over a period of four weeks. This included some amount of design time and all the implementation and debugging of the system, up to the point where it provided the functionality I described here.

I spent about two weeks developing and testing the program and the corresponding VIEWERS network. The remaining time was spent entering a small portion of the Chinese characters in the database and checking the system and the database for consistency.

Most of the time spent on the system after the initial development has been for upgrading the database, up to the size described in the previous section. I have also spent some time adding more functionality and fixing some bugs. The program, however, has remained substantially unchanged since the initial development. I have now used it almost daily for a couple of months, usually for about half an hour every day.

3.6 Performance

The current version of the Chinese Tutor, which was written in the simplest possible way and has never been optimized for performance, is well within acceptable response times for an interactive system. Its current performance, in fact, would be quite satisfactory even for a "polished" system.

Selection of operations, which the Tutor implements through the VIEWERS system, is quite efficient. The user selects an operation by moving through a network of frames, typically by typing single-key commands on a keyboard. The time to move from one frame to another is currently about 80 msec (more than 12 selections per second) and is quite adequate for interactive use. During rapid selection sequences the display of intermediate frames is suppressed by Hemlock's display routines.

Handling Chinese characters from the database is also quite efficient. The KR system is the main reason for the good performance, since it offers efficient primitive operations. As an example, retrieving a value or values from a KR schema requires only 52 microseconds. Each level of inheritance through the network of schemata requires an additional 12 microseconds. Such times are of the order of one to four function calls and are thus quite acceptable in terms of the overall system performance.

Sorting Chinese characters is among the most expensive operations. For instance, sorting all the characters by number of strokes currently takes 0.61 seconds. Sorting all characters by pronunciation, the most expensive operation since tone marks must be considered when sorting, takes about 1.1 seconds. Simple caching schemes could be used to eliminate the need for sorting almost completely.

The graphical display of Chinese characters through the VECTOR package is reasonably fast. This area, however, will be radically improved by planned changes to the Tutor (in particular, the conversion to bitmapped fonts). The current system can display a Chinese character in 140 msec, or about 7 characters per second. Preliminary estimates indicate that the introduction of bitmapped fonts will speed this up by at least a factor of 20, bringing the system into the 150 characters/second range.

4. Discussion

The previous section suggests an image of Common Lisp as a highly effective development environment. The program consists of about 2,000 lines of hand-written code and 1,500 lines of machine-generated code. It took me less than three man-weeks to develop and test the whole system.

One could argue that the rapid development time was only possible because of the availability of the tool-kit. This is certainly true, but, in my opinion, it actually reinforces the central thesis of this document: LISP is ideally suited for the rapid development of flexible, efficient software systems. Features of LISP such as polymorphism and program-as-data were essential ingredients to the construction of the system. Such features made it possible to quickly reuse all the pieces of the tool-kit without any modification.

The Common Lisp "package" structure provides a useful layer for a building-block approach such as the one I used. A very simple package structure (each tool lives in its own package and exports a simple functional interface) was adequate for the construction of the Chinese Tutor.

The interactive features typical of LISP were invaluable in the rapid development of the system. Several times I performed extensive changes to the system on-line, and the interactive environment made it possible to do this incrementally. After each change all internal data structures were available for inspection and modification if necessary. A non-interactive programming system would have required a major re-thinking and system-wide recompilation at every step.

Finally, the Chinese Tutor has been quite effective as a first evaluation vehicle for some of the tools developed by the Dante project. The Tutor is a highly interactive system that differs substantially from the mostly text-oriented, relatively static nature of applications that systems like ZOG and VIEWERS were originally developed for. Using VIEWERS as the main interface mechanism for the Tutor has exposed the need for a more dynamic behavior. As a result, I have already incorporated into the VIEWERS system facilities like recursive frame traversal and a general-purpose iteration mechanism.

The Tutor has also exposed the importance of both local and global control in determining the dynamic layout of complex presentation systems, where any piece of information on the screen may act as a handle through which the user can reach more detailed information. I am currently in the process of formalizing these ideas and incorporating them into the design of the Uniform Workstation Interface.

5. Future Work

I will now conclude with an outline of future directions of development for the Chinese Tutor. I will first indicate developments that are a logical extension of the current system and will be implemented within a short period of time. I will then present more ambitious plans that are likely to change radically the way the system looks to the user.

5.1 Short-Term Developments

Several improvements to the Chinese Tutor are already underway. First and foremost, I have begun the conversion from vector to bitmapped fonts. The new fonts will offer two advantages: speed and ease of construction.

Other short-term planned improvements include:

- more extensive on-line tutorials,
- support for alternative pronunciation systems (such as Wades-Giles), at least as a compatibility mode for students who learned Chinese with the older systems,
- a refined scoring mechanism that includes statistics on a student's characters confusion,[6]
- exploiting Hemlock's dynamic spelling checker to minimize typing errors when the student enters English text.

5.2 Long-Term Developments

A fundamental, long-term activity must be the development of the Chinese knowledge base. As in all knowledge-based systems, this portion is both time-consuming and error-prone. The current method is completely manual and is inappropriate for the development of a comprehensive knowledge base.

A more promising possibility seems to be the semi-automatic acquisition of knowledge from existing sources, such as books and dictionaries. However, very little work in this direction has been reported for language tutors, and especially Oriental language tutors.

Another type of extension will provide support for full sentences within the Chinese Tutor. The current structure of the knowledge representation is probably adequate, but much work remains to be done before the system can achieve the appropriate level of student support. Such an extension might conceivably require endowing the Tutor with a deeper understanding of Chinese grammar.

Another important extension to the system would be the addition of etymological information about Chinese characters and their derivation. Much more than in Western languages, appropriate knowledge of a Chinese character's etymology can greatly facilitate learning its meaning and its written form. Plentiful information exists in the literature, but it is not clear how such information could be mechanically transferred to a computer-based tutor.

Finally, the system should be extended in the direction of intelligent tutoring. After the initial period when learning to recognize characters is the main concern, all the usual considerations about Intelligent Tutoring Systems (see for instance [Carbonell 70] and [Barr and Feigenbaum 81]) become as essential in this domain as they are in any other intelligent tutoring system.

---

[6] It is often the case that beginners consistently mistake pairs of Chinese characters or pronunciations. The scoring mechanism could be extended to represent such pairwise errors and could focus on these particularly troublesome characters.

References

[Accetta *et al.* 86] Mike Accetta, Robert Baron, William Bolosky, David Golub, Richard Rashid, Avadis Tevanian, Michael Young.
Mach: A New Kernel Foundation for UNIX Development.
In *Proceedings of Summer Usenix.* July, 1986.

[Barr and Feigenbaum 81]
Avron Barr, Edward A. Feigenbaum (ed.).
The Handbook of Artificial Intelligence.
HeurisTech Press, Stanford, California, 1981.

[Carbonell 70] Jaime R. Carbonell.
AI in CAI: An Artificial-Inteliigence Approach to Computer-Assisted Instruction.
IEEE Transactions on Man-Machine Interface MMS-11(4):190-202, December, 1970.

[Carnegie Group 86]
Knowledge Craft Reference Manual
Carnegie Group, Inc., Pittsburgh, PA, 1986.

[Fox *et al.* 84] M.S.Fox, J.Mark Wright, D.Adam.
Experiences with SRL: An Analysis of a Frame-Based Knowledge Representation.
In *First International Workshop on Expert Database Systems.* 1984.

[Giuse 86] Dario Giuse.
Research in Uniform Workstation Interfaces - Research Proposal to DARPA
1986.

[Giuse 87] Dario Giuse.
KR: Efficient Knowledge Representation.
Technical Report, Robotics Institute, Carnegie-Mellon University, to appear in 1987.

[Giuse *et al.* 83] D. Giuse, D.P. Siewiorek, W.P. Birmingham.
DEMETER: A Design Methodology and Environment.
Technical Report CMUCAD-83-14, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1983.

[Hershey 76] Norman M. Wolcott, Joseph Hilsenrath.
A Contribution to Computer Typesetting Techniques.
National Bureau of Standards, Washington, D.C. 20234, 1976.

[MacLachlan 84] Rob MacLachlan.
Hemlock Command Implementor's Manual.
Technical Report Spice S177, Carnegie-Mellon University, August, 1984.

[MacLachlan 86] Rob MacLachlan.
Hemlock User's Manual.
Technical Report Spice S178, Carnegie-Mellon University, March, 1986.

[McCarthy 60] John McCarthy.
Recursive Functions of Symbolic Expressions and their Computation by Machine.
Comm. ACM 3(4):184-195, April, 1960.

[Robertson, McCracken, and Newell 80]
G. Robertson, D. McCracken, A. Newell.
The ZOG approach to Man-Machine Communication.
International Journal of Man-Machine Studies 14:461-488, 1980.

[Steele 84] Guy L. Steele.
Common LISP - The Language.
Digital Press, Burlington, MA, 1984.

[Tevanian 86] Avadis Tevanian, Jr.
Mach: A New Basis for Future UNIX Development.
In *Proceedings of Autumn EUUG*. September, 1986.

[Weinreb and Moon 81]
Weinreb, D. and Moon, D.
Lisp Machine Manual
Symbolics, Inc., Cambridge, MA, 1981.

[Wright and Fox 83]
M. Wright, M. Fox.
SRL: Schema Representation Language.
Technical Report, Carnegie-Mellon University, December, 1983.

[XSIS 85] Xerox Special Information Systems.
NoteCards Release 1.2i Reference Manual
Xerox Special Information Systems, Pasadena, California, 1985.